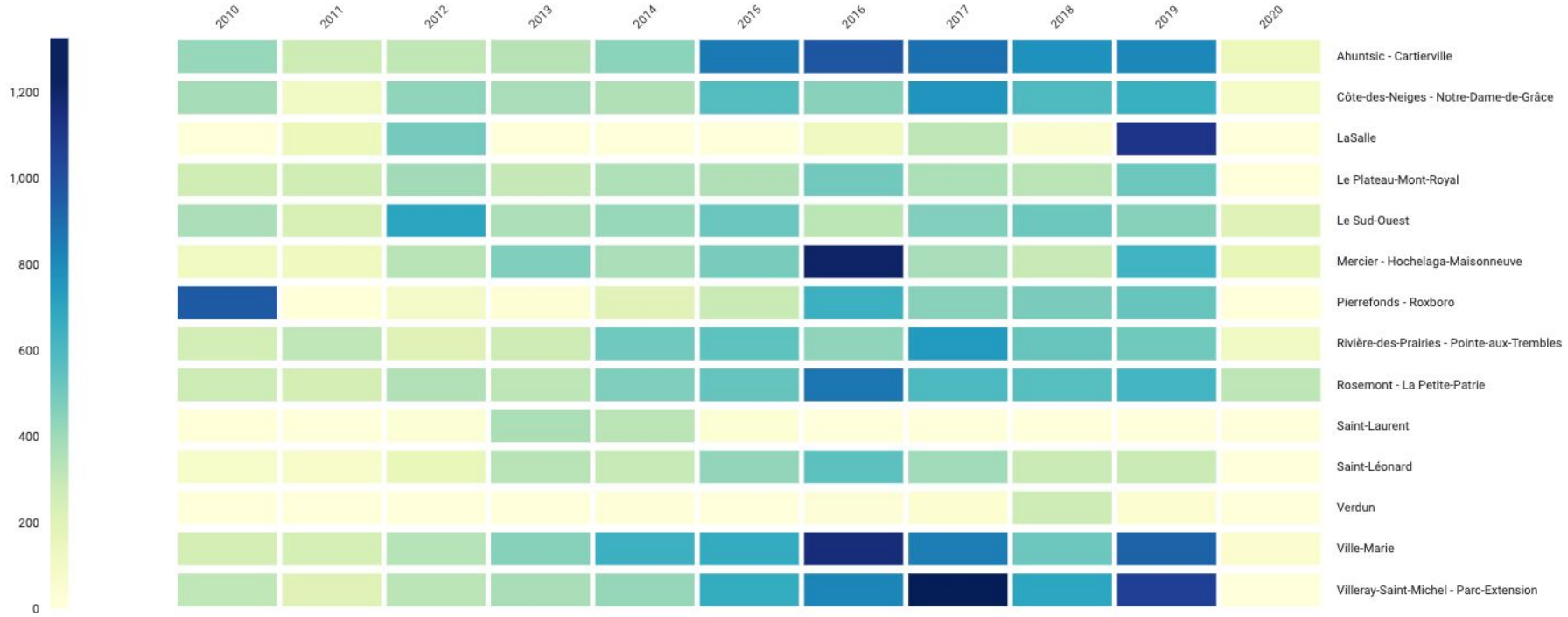

# Guide TP3

## INF8808 | Summer 2022

**Version JavaScript**

# Goals

- Create an interactive heatmap with hover feature



# Data

## *Trees planted in Montreal*

- The data represents trees planted over time in Montreal's neighborhoods
- File : `src/assets/data/arbres.csv`
- Columns :
  - **Arrond :** neighborhood ID
  - **Arrond_Nom:** neighborhood name
  - **Date_Plantation :** date of tree plantation
  - **Longitude**
  - **Latitude**

**Exemple** :

```
Arrond,Arrond_Nom,Date_Plantation,Longitude,Latitude

1,Ahuntsic - Cartierville,2004-06-10,-73.715515,45.535151
```

# Data preprocessing

***Goal** : Reorganize certain parts of it so they can be properly used by the D3 library*

**File** : `preprocessing.js`

4 functions to complete :

1. ***getNeighborhoodNames***
2. ***filterYears***
3. ***summaryYearlyCounts***
4. ***fillMissingData***

# Data preprocessing

## *Result*

- Example :

```
[
    {
        "Arrond_Nom": "Ville-Marie",
        "Plantation_Year": 2017,
        "Comptes": 888
    },
    {
        "Arrond_Nom": "...",
        "Plantation_Year": ...,
        "Comptes": ...
    },
    ...
]
```

# Data preprocessing

## *Remark*

***We provide a function*** `range` ***that could be useful*** *(optional)*

```
/**
 * Utilitary funtion that returns an array of number in the given range, inclusively.
 *
 * @param {number} start The starting number
 * @param {number} stop The end number
 * @returns {number[]} The array with a sequence of numbers within the given range
 *
 */
export function range (start, stop) {
  const res = []
  for (var i = start; i <= stop; i++) {
    res.push(i)
  }
  return res
}
```

```
/**
 * For the heat map, fills empty values with zeros where a year is missing for a neighborhood because
 * no trees were planted or the data was not entered that year.
 *
 * @param {object[]} data The datas set to process
 * @param {string[]} neighborhoods The names of the neighborhoods
 * @param {number} start The start year (inclusive)
 * @param {number} end The end year (inclusive)
 * @param {Function} range A utilitary function that could be useful to get the range of years
 * @returns {object[]} The data set with a new object for missing year and neighborhood combinations,
 * where the values for 'Counts' is 0
 */
export function fillMissingData (data, neighborhoods, start, end, range) {
  // TODO : Find missing data and fill with 0

 // EXAMPLE :

  const myRange = range(6,15)

  // myRange is : [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

  return []
}
```

# Heatmap

## *5 steps*

- In the file `vis.js` :
  - ***setColorScaleDomain***
    - *For the color of the rectangles*
  - ***appendRects***
    - *Add the SVG rects to be displayed*
  - ***updateXScale*** & ***updateYScale***
    - *To help set the position and size of the rects*
  - ***drawXAxis***, ***drawYAxis*** & ***rotateXTicks***
  - ***updateRects***
    - *Display the rects with correct color, size, placement, etc.*

# Heatmap

## *Rectangles traced in 2 steps*

### 1. Add ‘<g>’ elements containing ‘<rect>’ elements

```
/**
 * For each data element, appends a group 'g' to which an SVG rect is appended
 *
 * @param {object[]} data The data to use for binding
 */
export function appendRects (data) {
  // TODO : Append SVG rect elements
}
```

### 2. Selection of the rectangles and setting the position, size and color

```
/**
 * After the rectangles have been appended, this function dictates
 * their position, size and fill color.
 *
 * @param {*} xScale The x scale used to position the rectangles
 * @param {*} yScale The y scale used to position the rectangles
 * @param {*} colorScale The color scale used to set the rectangles' colors
 */
export function updateRects (xScale, yScale, colorScale) {
  // TODO : Set position, size and fill of rectangles according to bound data
}
```

Remarks :

- xScale and yScale are both **d3.scaleBand**
- colorScale is **d3.scaleSequential**

# Legend

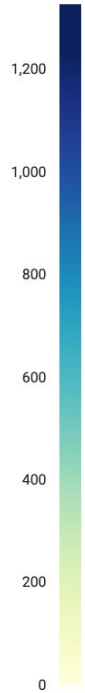
- Displayed at the left of the graph

- **File :** `legend.js`

- The provided code already accomplishes part of the tasks -- it :
  - Initializes the gradient to be used
  - Initializes the rectangle that will represent the legend
  - Initializes the group for the axis

- Your task will be to manipulate these existing elements to display the legend

*Result :*



# Legend

## *Fill with SVG gradient*

The fill of the legend's rectangle is using an SVG gradient.

See this example for help.

- From https://developer.mozilla.org/en-US/docs/Web/SVG/Element/linearGradient

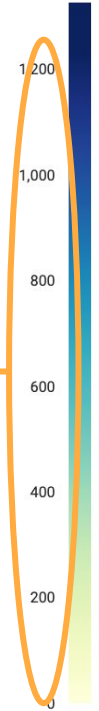
```
<svg viewBox="0 0 10 10" xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <defs>
    <linearGradient id="myGradient" gradientTransform="rotate(90)">
      <stop offset="5%"  stop-color="gold" />
      <stop offset="95%" stop-color="red" />
    </linearGradient>
  </defs>

  <!-- using my linear gradient -->
  <circle cx="5" cy="5" r="4" fill="url('#myGradient')" />
</svg>
```

# Legend

- To trace the axes you cannot directly use the color scale

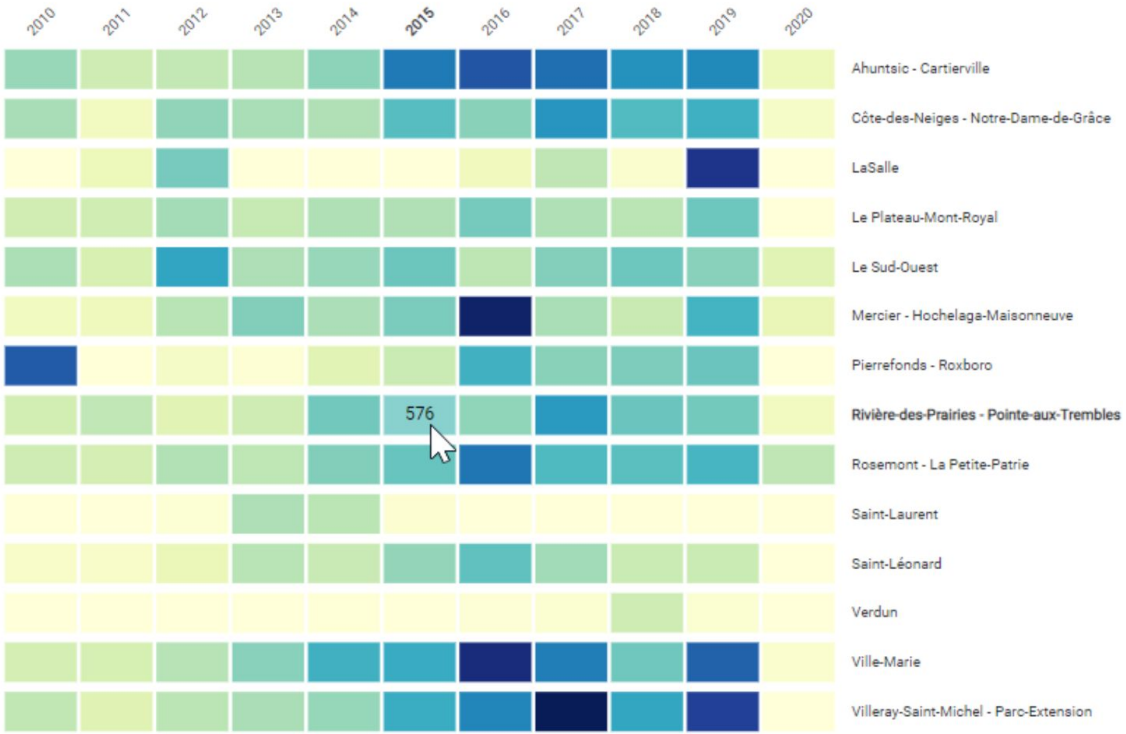
```
/**
 * Draws the legend to the left of the graphic.
 *
 * @param {number} x The x position of the legend
 * @param {number} y The y position of the legend
 * @param {number} height The height of the legend
 * @param {number} width The width of the legend
 * @param {string} fill The fill of the legend
 * @param {*} colorScale The color scale represented by the legend
 */
export function draw (x, y, height, width, fill, colorScale) {
  // TODO : Draw the legend
}
```

1. **Use a D3 scale that will allow to convert these continuous values to positions**
2. **Trace the D3 axis**

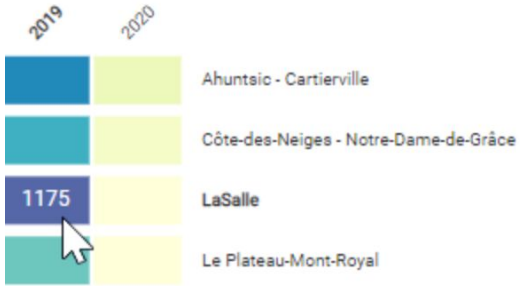


# Hover

*Illustrated*




***Remark :The count is black or white if the value is 1000 or more***



# Hover

- In `hover.js` :

    1. ***setRectHandler***
    2. ***rectSelected*** & ***rectUnselected***
        - The rectangle's opacity is 75% when hovered and 100% otherwise
    3. While a rectangle is hovered, the corresponding ticks should be bold, in ***selectTicks*** & ***unselectTicks***
        - **Please note that bold might look different on different browsers and devices**

# Text positioning

*In the axis and hover feature*

Maybe useful links :

- Transform
  - https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/transform
- Text-anchor
  - https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/text-anchor
- Dominant-baseline
  - https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/dominant-baseline
- Etc.

# Due date

Submission : May 29 11:59PM